



(11)

EP 2 256 639 A1

(12) **EUROPEAN PATENT APPLICATION**

(43) Date of publication:
01.12.2010 Bulletin 2010/48

(51) Int Cl.:
G06F 13/16 (2006.01)

(21) Application number: **10163389.9**

(22) Date of filing: **20.05.2010**

(84) Designated Contracting States:
AL AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HR HU IE IS IT LI LT LU LV MC MK MT NL NO PL PT RO SE SI SK SM TR
Designated Extension States:
BA ME RS

(72) Inventors:
• **Ryu, Seung-Woo**
Seoul (KR)
• **Ryu, Jae-Min**
Gyeonggi-do (KR)
• **Lee, Chul**
Gyeonggi-do (KR)

(30) Priority: **22.05.2009 KR 20090045118**

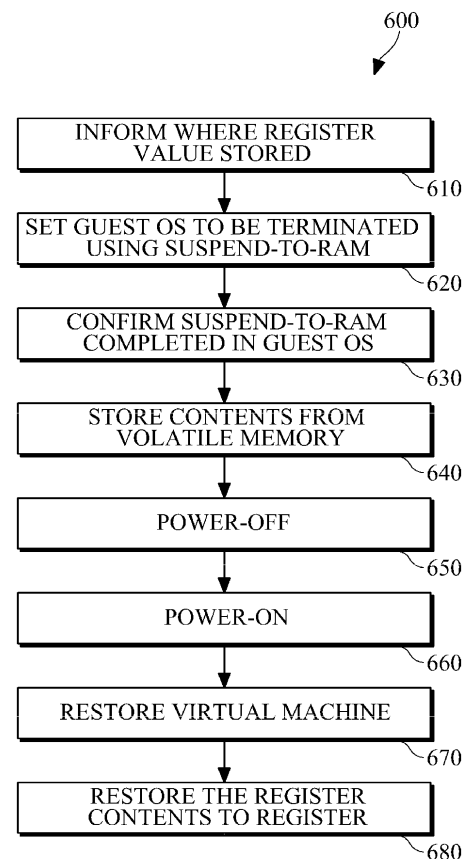
(71) Applicant: **Samsung Electronics Co., Ltd.**
Suwon-si,
Gyeonggi-do 442-742 (KR)

(74) Representative: **D'Halleweyn, Nele Veerle Trees**
Gertrudis et al
Arnold & Siedsma
Sweelinckplein 1
2517 GK Den Haag (NL)

(54) **Memory apparatus and method therefor**

(57) A memory apparatus and an operation of the memory apparatus which allow quick booting are provided. The memory apparatus includes a volatile memory, a non-volatile memory, and a memory control unit to control input/output of data stored in the volatile memory and the non-volatile memory. The memory control unit restores data, according to a control command input from outside of the memory apparatus, from the non-volatile memory to the volatile memory in an on-demand fashion during booting.

FIG. 6



EP 2 256 639 A1

Description

[0001] The invention relates to a memory apparatus and a method for the memory apparatus, and more particularly, to a memory apparatus enabling quick booting and a method for the memory apparatus.

[0002] A computer system processor executes commands from a volatile memory such as a dynamic random access memory (DRAM). DRAM is capable of high speed access, but loses data stored therein when power is turned off since it is a volatile memory. Thus, data held in DRAM needs to be stored in a non-volatile memory such as hard disk and flash memory before the power supply to the DRAM is stopped.

[0003] For example, in a personal computer, default programs may be written in DRAM in order for the DRAM to exchange data with a keyboard or hard disk since no information is stored in the DRAM immediately after the power is supplied. This start-up process of the computer or initialization process of settings is referred to as "booting." Generally, the default programs are stored in storage mediums such as read-only memory (ROM) or hard disk drive (HDD). Data transmission from the above storage medium to DRAM occurs at low speed. Hence, a waiting time for a computer to boot when it is turned on is long and it takes a significant amount of time to initialize hardware.

[0004] The object of the invention is to provide a memory apparatus and a method for the memory apparatus, which can reduce booting time of a computing device.

[0005] In one general aspect, there is provided a memory apparatus including a volatile memory, a non-volatile memory, and a memory control unit. The memory control unit restores data, according to an externally input signal, from the non-volatile memory to the volatile memory in an on-demand fashion during booting. According to a preferred aspect the volatile memory may be a dynamic random access memory (DRAM). According to another preferred aspect the non-volatile memory may comprise at least one of NOR flash memory, NAND flash memory, and phase-change random access memory (PCRAM).

[0006] The memory control unit may determine whether data according to an address and a control signal which are included in the input signal is present in the volatile memory and, if the data is not present in the volatile memory, restore the data from the non-volatile memory to the volatile memory. A look-up table which includes information indicating whether data may be present at corresponding addresses of the volatile memory may be used in order to determine whether the data is present in the volatile memory. The look-up table may be a hardware component to expedite the determination process. The hardware component may be a flip-flop.

[0007] When it is determined that data corresponding to the address included in the input signal is present in the volatile memory and the control signal is a read signal, the memory control unit may transfer the externally input signal to the volatile memory such that the data is output

from the volatile memory.

[0008] The memory control unit may control information stored in a single volatile memory to be stored in a plurality of corresponding non-volatile memories through multiple channels in a distributed manner in response to a zero power suspend-to-RAM (ZPSTR) mode command, wherein in ZPSTR mode, an operation of storing data stored in the volatile memory into the non-volatile memory may be performed. A mode command may be provided as a physical signal standardized in a predetermined form. The physical signal may be an electrical signal or a register command of the memory itself which instructs a specific register of command registers to output a command. Therefore, compatibility may be provided since the memory apparatus can be used without changing the existing hardware interface, and it may be possible to partially change and use a suspend-to-RAM (STR) mode which is used by existing operating systems (OS).

[0009] The memory control unit may store the data stored in the volatile memory into the non-volatile memory using a memory mapping table which shows a mapping relationship between a storage area of the volatile memory and a storage area of the non-volatile memory.

[0010] In addition, the memory control unit may control pieces of data related to the same application to be stored as closely together as possible in a particular storage area of the non-volatile memory in response to a storage request command from an operating system operated outside of the memory apparatus to request the data related to the same application from among the data of the volatile memory to be stored into the non-volatile memory.

[0011] Moreover, the memory control unit may store data related to a predetermined application in the non-volatile memory regardless of the input of a ZPSTR mode command in response to a request from outside of the memory apparatus to store the data related to the predetermined application from among data of the volatile memory into the non-volatile memory, wherein, in ZPSTR mode, an operation of storing data stored in the volatile memory into the non-volatile memory is performed.

[0012] Furthermore, the memory control unit may swap out some of the data stored in the volatile memory to the non-volatile memory in response to a swap command from an operating system operated outside of the memory apparatus.

[0013] Also, the memory control unit may include a decoder, a restoration determining unit, and a flow control unit. The decoder decodes an address and a control signal which are externally input. The restoration determining unit determines whether data requested according to the address and the control signal is present in the volatile memory. The flow control unit controls the input/output of data stored in the volatile memory and the non-volatile memory in accordance with the control signal. Additionally, the memory control unit may restore the data from the non-volatile memory to the volatile memory and then

erase the restored data from the non-volatile memory.

[0014] In another general aspect, there is provided a method for a memory apparatus which includes a volatile memory, a non-volatile memory, and a memory control unit to control input/output of data stored in the volatile memory and the non-volatile memory, the method including restoring data, according to an externally input signal, from the non-volatile memory to the volatile memory in an on-demand fashion during booting.

[0015] According to a preferred aspect of the method, the volatile memory comprises a dynamic random access memory (DRAM).

[0016] According to a preferred aspect of the method, the non-volatile memory comprises at least one of NOR flash memory, NAND flash memory, and phase-change random access memory (PCRAM).

[0017] According to a preferred aspect of the method, the restoring of the data to the volatile memory comprises determining whether data, according to an address and a control signal which are included in the input signal, is present in the volatile memory; and if the data is not present in the volatile memory, restoring the data from the non-volatile memory to the volatile memory.

[0018] According to a preferred aspect of the method, the determining of whether the data is present in the volatile memory uses a look-up table which comprises information indicating whether data is present at corresponding addresses of the volatile memory.

[0019] According to a preferred aspect the method further comprises, when it is determined that data corresponding to the address included in the input signal is present in the volatile memory and the control signal is a read signal, transferring the externally input signal to the volatile memory such that the data is output from the volatile memory.

[0020] According to a preferred aspect the method further comprises storing information stored in a single volatile memory into a plurality of corresponding non-volatile memories through multiple channels in a distributed manner in response to a zero power suspend-to-RAM (ZPSTR) mode command, wherein, in ZPSTR mode, operation of storing data stored in the volatile memory into the non-volatile memory is performed.

[0021] According to another aspect of the invention, there is provided a method for switching between a virtual machine and a guest operating system (OS) in a computer, the computer configured to support a suspend-to-RAM function, the method comprising: informing the virtual machine of an area in a volatile memory in which contents of a register, corresponding to the to the guest OS, is stored from the register, the volatile memory comprising at least one area; setting the guest OS to be terminated using suspend-to-RAM; confirming, by the virtual machine, that suspend-to-RAM is completed in the guest OS; storing, by the virtual machine, all contents of the area in the volatile memory; performing a power-off operation; performing a power-on function; restoring the virtual machine; and restoring the register contents to the

register from the area in the volatile memory.

[0022] According to a preferred aspect, the method further comprises partitioning the area in volatile memory into a plurality of areas respectively corresponding to the guest OS and the virtual machine.

[0023] According to a preferred aspect, the method further comprises storing data in a general storage unit using a suspend-to-disk function of the virtual machine.

[0024] According to a preferred aspect, the method further comprises in response to an instruction being executed in the computer, memory address access is requested.

[0025] According to a preferred aspect, the method further comprises passing the access through a pseudo physical memory layer of the virtual machine using a page fault.

[0026] According to a preferred aspect, the method further comprises managing, by the virtual machine, restored and non-restored areas with a table.

[0027] According to a preferred aspect, the method further comprises managing, by the virtual machine, restored and non-restored areas of the volatile memory with a bitmap.

[0028] According to a preferred aspect, the method further comprises, in response to memory access to a non-restored area occurring, reading the area on a block-by-block basis.

[0029] According to a preferred aspect, the method further comprises restoring the non-restored area.

[0030] According to another aspect of the invention, there is provided a computer supporting a suspend-to-RAM function, the computer comprising: a central processing unit (CPU) comprising a register, the register comprising contents; a volatile memory comprising at least one area; a non-volatile memory; a virtual machine; a guest operating system (OS) being configured to be terminated by suspend-to-RAM; and a memory controller configured to: switch between the virtual machine and the guest (OS); and inform the virtual machine of an area in a volatile memory in which contents of a register, corresponding to the to the guest OS, are stored from the register, wherein the virtual machine is configured to: confirm that suspend-to-RAM is completed in the guest OS, store all contents of the area in the volatile memory, wherein the computer is configured to: perform a power-off operation, and performing a power-on function, wherein the memory controller is further configured to: restore the virtual machine, and restore the register contents to the register from the area in the volatile memory.

[0031] According to a preferred aspect, the memory controller is further configured to partition the area in volatile memory into a plurality of areas respectively corresponding to the guest OS and the virtual machine.

[0032] According to a preferred aspect, the computer further comprises a general storage unit configured to store data using a suspend-to-disk function of the virtual machine.

[0033] According to a preferred aspect, the computer

is further configured to execute and instruction to request memory address access.

[0034] According to a preferred aspect, the virtual machine is further configured to pass the access through a pseudo physical memory layer of the virtual machine using a page fault.

[0035] According to a preferred aspect, the virtual machine is further configured to manage restored and non-restored areas of the volatile memory with a table.

[0036] According to a preferred aspect, the virtual machine is further configured to manage restored and non-restored areas of the volatile memory with a bitmap.

[0037] According to preferred aspect, the memory controller is further configured to, in response to memory access to a non-restored area occurring, read the area on a block-by-block basis.

[0038] According to a preferred aspect, the memory controller is further configured to restore the non-restored area.

[0039] Other features and aspects will be apparent from the following detailed description, the drawings, and the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[0040]

FIG. 1 is a diagram illustrating an example computing system including a memory apparatus.

FIG. 2 is a diagram illustrating an example of the memory apparatus of FIG. 1.

FIG. 3 is a diagram illustrating the memory apparatus of FIG. 2.

FIG. 4 is a flowchart illustrating an example of a method for a memory apparatus.

FIG. 5 is a flowchart illustrating another example of a method for a memory apparatus.

FIG. 6 is a flowchart illustrating another example of a method according to an embodiment.

[0041] Elements, features, and structures are denoted by the same reference numerals throughout the drawings and the detailed description, and the relative size and proportions of some elements may be exaggerated in the drawings for clarity, illustration, and convenience.

DETAILED DESCRIPTION

[0042] The following detailed description is provided to assist the reader in gaining a comprehensive understanding of the methods, apparatuses and/or systems described herein. Accordingly, various changes, modifications, and equivalents of the systems, apparatuses

and/or methods described herein will suggest themselves to those of ordinary skill in the art. The progression of processing steps and/or operations described is an example; however, the sequence of and/or operations is not limited to that set forth herein and may be changed as is known in the art, with the exception of steps and/or operations necessarily occurring in a certain order. Descriptions of well-known functions and structures may be omitted for increased clarity and conciseness.

[0043] FIG. 1 is a block diagram illustrating an example computing system 1 including a memory apparatus 200.

[0044] The computing system 1 includes a central processing unit (CPU) 100 and the memory apparatus 200. The CPU 100 may include at least one register 105. The computing system 1 executes a series of pre-written commands such as programs or program modules which may operate in a predetermined scheme. The program modules may include routines, programs, objects, components, and data configurations, which may perform particular tasks or implement particular abstract data types. The computing system 1 may be an electronic device in various forms, such as a personal computer, a laptop, and an embedded system, and there is no limitation in its type and form.

[0045] The CPU 100 may perform various operations and controls. In order to input and output data to/from the memory apparatus 200, the CPU 100 may transmit to the memory apparatus 200 a control signal 10, which includes a read/write command, and an address, which indicates a storage location of the data, and may transmit and receive data to/from the memory apparatus 200.

[0046] The memory apparatus 200 includes a memory control unit 210, a volatile memory 220, and a non-volatile memory 230. The memory control unit 210 may control data input/output between the CPU 100 and the volatile memory 220 and non-volatile memory 230. The memory control unit 210 may function as a buffer which is used to restore data from the non-volatile memory 230 to the volatile memory 220, and may include a random access memory (RAM; not shown) that stores data for operation of the memory control unit 210. Examples of RAM may include static random access memory (SRAM).

[0047] Although FIG. 1 illustrates that the memory apparatus 200 includes one volatile memory 220 and one non-volatile memory 230, a number of volatile memories 220 and non-volatile memories 230 may be provided to write and read data in parallel manner.

[0048] The volatile memory 220 may be used as a system memory to temporarily store data processing result of the computing system 1. The volatile memory 220 may be a dynamic random access memory (DRAM). Other types of volatile memory may also be employed as the volatile memory 220, for example, SDRAM, SRAM, etc. The list is nonlimiting. The non-volatile memory 230 may preserve data even when the power is turned off and may be used to back up data stored in the volatile memory 220. The non-volatile memory 230 may be a NOR flash memory, a NAND flash memory, or a phase-change

memory (PCRAM). Other types of non-volatile memory may also be employed as the non-volatile memory 230. The list is nonlimiting.

[0049] An operating system may collect as much memory as possible without distributing memory areas corresponding to applications, and may write data in the non-volatile memory.

[0050] Although not illustrated in FIG. 1, the computing system 1 may be configured to include an external power switch, a power-supply, and an internal power switch that collectively control the power supply to each of the memory control unit 210, the volatile memory 220, and the non-volatile memory 230.

[0051] The computing system 1 may operate in normal mode, in suspend-to-RAM (STR) mode, and in zero-power-suspend-to-RAM (ZPSTR) mode. In addition, the computing system 1 may perform normal booting in which the system initialization is executed when the power is turned on, suspend-to-RAM mode releasing booting to switch from suspend-to-RAM to normal mode, and ZPSTR mode releasing booting to switch from ZPSTR to normal mode. The computing system 1 may use, for example, a plurality of booting programs stored in a predetermined area of the non-volatile memory 230 or an individual ROM to execute the normal booting, the suspend-to-RAM mode releasing booting, or the ZPSTR mode releasing booting according to a situation.

[0052] A mode command to instruct the computing system 1 to enter a particular mode or a command to release the computing system 1 from a particular mode, which is input from the CPU 100, may be provided as a physical signal standardized in a predetermined form. The physical signal may be an electrical signal or a register command of the memory itself which instructs a specific register of command registers to output a command. Therefore, compatibility may be provided since the memory apparatus 200 can be used without changing the existing hardware interface, and it may be possible to partially change and use a suspend-to-RAM mode which is used by existing operating systems (OS).

[0053] Operation of the normal booting of the computer system 1 will be described below.

[0054] When the power is turned on, a basic input output system (BIOS) booting program stored in the predetermined area of the non-volatile memory may be transferred to the volatile memory to allow the CPU to perform a booting operation. Furthermore, the memory control unit may perform operations for initializing the computing system 1 by controlling the volatile memory and the non-volatile memory, such that data is restored from the non-volatile memory to the volatile memory according to the order defined by BIOS under the control of the CPU.

[0055] Hereinafter, a suspend-to-RAM mode operation of the computing system 1 will be described.

[0056] In the suspend-to-RAM mode, power may be turned off to the CPU 100, but is supplied to the volatile memory 220. When receiving a suspend-to-RAM mode command as a control signal 10 from the CPU 100, the

memory control unit 210 may store a CPU register value to the volatile memory 220. The CPU register value may be a value of an internal register, which may record a value used by the CPU, which may be an important value. For example, the CPU register value may include a program counter which indicates a location of a command currently executed on the memory.

[0057] In this case, when a suspend-to-RAM mode releasing command is input from the CPU 100, the memory control unit 210 may restore the CPU register value stored in the volatile memory 220 to the CPU 100, allowing the CPU 100 to execute a subsequent command.

[0058] Hereinafter, a ZPSTR mode operation of the computing system 1 will be described.

[0059] In the ZPSTR mode, power supply to the CPU 100 and to the non-volatile memory 220 may be disconnected. Accordingly, when a ZPSTR mode command is input from the CPU 100, the memory control unit 210 may copy the CPU register value and data stored in the volatile memory 220 to the non-volatile memory 230. For example, the CPU register value may be stored in a predetermined area of the non-volatile memory 230.

[0060] To quickly enter ZPSTR mode, data of a single volatile memory 220 may be stored in a plurality of non-volatile memories 230 through multiple channels in a parallel manner. If multiple volatile memories 220 are present and a corresponding number of non-volatile memories 230 are present, data of the volatile memories 220 may be, respectively, stored in the non-volatile memories 230 in a distributed manner.

[0061] To store data of the volatile memory 220 into a plurality of non-volatile memories 230 effectively, a memory mapping table may be further included, which represents a mapping relationship between storage areas of the respective non-volatile memories 230 and a corresponding storage area of the volatile memory 220. For example, the memory mapping table may indicate a one-to-one mapping relationship between the storage area of the volatile memory 220 and a part of the storage area of each non-volatile memory 230, except for an area where, for example, BIOS or flash translation layer (FTL) data is stored. The memory mapping table may be stored in a particular area in each non-volatile memory 230, and the memory control unit 210 may read the memory mapping table when the data of the volatile memory 220 is to be stored in the non-volatile memories 230. The memory mapping table may be included in the FTL.

[0062] In response to a ZPSTR releasing booting command from the CPU 100, the memory control unit 210 may restore the CPU register value stored in a specific area of the non-volatile memory 230 to the CPU 100, and may restore data of the non-volatile memory 230 to the volatile memory 220. In one example, not all the data previously copied from the non-volatile memory 230 to the volatile memory 220 is restored, but data according to a restored program counter included in the CPU register may only be restored from the non-volatile memory 230 to the volatile memory 220 in an on-demand fashion.

[0063] During data restoration in an on-demand fashion, the memory control unit 210 may determine whether data which the CPU 100 requests to read is present in the volatile memory 220. To determine whether the requested data is present in the volatile memory 220, e.g., whether the requested data is available, the memory control unit 210 may be configured to include a restoration determining unit 214 which may be a hardware component as shown in FIG. 2.

[0064] In one implementation, since the memory control unit 210 as a hardware device may copy data from the volatile memory 220 to the non-volatile memory 230, restore the data from the non-volatile memory 230 to the volatile memory 220, and perform booting using hardware components in an on-demand fashion, the shift between ZPSTR mode and normal mode can be carried out more quickly than when performed under the control of software or an operating system.

[0065] It may be more efficient to sequentially read relevant data which has been written in an adjacent memory from the non-volatile memory 230 than to read data from distributed memory areas. In one implementation, when a command for requesting to store data which is related to the same application and stored in the volatile memory 220 to the non-volatile memory 230 is input from an operating system operated outside of the memory apparatus 200, for example, an operating system operated in the CPU 100, the memory control unit 210 may control the data related to the same application to be stored as closely together as possible in a specific area of the non-volatile memory 230.

[0066] In addition, in response to a request from an operating system operated outside of the memory apparatus 200 for storing data related to a given application from among data stored in the volatile memory 220 into the non-volatile memory 230, the memory control unit 230 may store the corresponding data related to the given application from the volatile memory 220 into the non-volatile memory 230 regardless of a ZPSTR mode command by which data stored in the volatile memory 220 is stored into the non-volatile memory 230. For example, when the ZPSTR mode command is input later on, the data previously stored into the non-volatile memory 230 may not need to be stored again. Therefore, data in the volatile memory 220, except for the previously stored data, may be stored into the non-volatile memory 230, and accordingly the storing time is reduced so that quick entering into the ZPSTR mode is possible.

[0067] To overcome physical capacity limitations of a system memory, a virtual memory scheme may be used to utilize a swap device or a swap partition, which may have a large capacity, as a part of a volatile memory. In one implementation, the memory control unit 210 may use the non-volatile memory 230 for a virtual memory scheme. For example, in response to a swap command from the operating system (OS), the memory control unit 210 may swap out some data which is not immediately used by the volatile memory 220 to the non-volatile mem-

ory 230, and swap data in the volatile memory 220 from the non-volatile memory 230 when the data swapped out to the non-volatile memory 230 is needed.

[0068] FIG. 2 is a block diagram illustrating an example of the memory apparatus of FIG. 1.

[0069] The memory apparatus 200 may include a decoder 212, the restoration determining unit 214, a flow control unit 216, the volatile memory 220 and the non-volatile memory 230. Configurations of the decoder 212, the restoration determining unit 214 and the flow control unit 216 correspond to the configuration of the memory control unit 210.

[0070] The decoder 212 may decode a control signal 10 and an address signal 20, which indicate commands input from the CPU 100.

[0071] The restoration determining unit 214 may determine whether data corresponding to an address input from the CPU 100 is present in the volatile memory 220. The restoration determining unit 214 may utilize a look-up table including information which indicates whether data corresponding to each address is present in the volatile memory 220.

[0072] The look-up table may be configured as a hardware component. As an example, the look-up table may be configured as a flip-flop. In this case, storage units of blocks of the non-volatile memory 230 may be mapped one to one to storage units of blocks of the volatile memory 220, and the flip-flop may be configured to indicate whether data of a corresponding unit block is present in the volatile memory 220 or the non-volatile memory 230. For example, the look-up table may include a plurality of blocks if the memory is divided into blocks, or may include a plurality of pages when each block is divided into units of pages. That is, there may be no limitation in the size of the corresponding memory areas of the volatile memory 220 and the non-volatile memory 230 for data storage. Also, software may be used in place of the flip-flop, as well as other hardware options to perform a corresponding function.

[0073] The flow control unit 216 may control the input/output of data stored in the volatile memory 220 and the non-volatile memory 230 in response to a control signal. The control signal may be generated automatically, or in response to a user input. For example, if the user selects or executes a STR or ZPSTR mode (e.g., in software), it can generate the control signal. In addition, if the user pushes the computer's power-off button, the computer can cause a hardware interrupt and then an operating system (OS) can generate the control signal. If the restoration determining unit 214 determines that data corresponding to an input address is not present in the volatile memory 220 and the control signal 10 is a read signal, the flow control unit 216 may generate a signal for copying data in a memory area (e.g., a page) corresponding to an address input from the non-volatile memory 230 to the volatile memory 220 and restore the data from the memory area to the volatile memory 220.

[0074] Moreover, the flow control unit 216 may erase

the data restored from the non-volatile memory 230 to the volatile memory 220 after completing the data restoration. The erase operation may not need to be performed at each restoration operation, and it may be possible to be performed while a normal operation is carried out under the control of the CPU 100. If a read command is received during the erase operation, the flow control unit 216 may suspend the erase operation and execute the read command. Alternatively, the data may be erased all together from the non-volatile memory 220 upon entering ZPSTR mode, and data of the volatile memory 220 may be written in the non-volatile memory 230 at the next entry into ZPSTR mode.

[0075] If the restoration determining unit 214 determines that data corresponding to an input address is present in the volatile memory 220 and the control signal is a read signal, the flow control unit 216 may control a signal input from outside to be transmitted to the volatile memory 220.

[0076] If the control signal is a write signal, data 30 input from the CPU 100 may be transmitted to the volatile memory 220.

[0077] Additionally, the flow control unit 216 may perform data write and read operations sequentially when copying data from the volatile memory 220 to the non-volatile memory 230 and restoring the data from the non-volatile memory 230 to the volatile memory 220. In one example, since random reading occurs in the read operation, the flow control unit 216 may induce a sequential read operation. Particularly, the flow control unit 216 may perform data writing such that pieces of data related to a single application are not written in the non-volatile memory 230 in a distributed manner, reducing time to restore data to the volatile memory 220.

[0078] FIG. 3 illustrates an example of the memory apparatus of FIG. 2.

[0079] The memory apparatus 200 may include the decoder 212, the restoration determining unit 214, a mode generating unit 310, a first arbiter 320, a second arbiter 330, the volatile memory 220, and the non-volatile memory 230. The elements denoted by the same reference numeral as the elements in FIG. 2 have the same configuration and perform the same operation, and thus the detailed description thereof will not be reiterated. The mode generating unit 310, the first arbiter 320, and the second arbiter 330 correspond to the configuration of the flow control unit 216 of FIG. 2.

[0080] The restoration determining unit 214 may determine, using a look-up table 215, whether requested data is present in the volatile memory 220. The look-up table 215 may be configured to include an "I" bit in each address for indicating the availability of data in the volatile memory 220. For example, if an "I" bit of an address is 0, it may indicate that data corresponding to the address is available in the volatile memory 220; and if an "I" bit of the address is 1, it may indicate that data corresponding to the address is not available in the volatile memory 220. Therefore, since there is no available data in the volatile

memory immediately after ZPSTR releasing booting is performed, all "I" bits in the look-up table 215 may be set to 1 as default.

[0081] The mode generating unit 310 may receive "I" bit information of an address from the restoration determining unit 214, and control the output/input of data stored in the volatile memory 220 and the non-volatile memory 230 according to a result of decoding a control signal 10 by the decoder 212. The mode generating unit 310 may generate a signal for copying a page from the non-volatile memory 230 to the volatile memory 220 when an "I" bit of the look-up table is 1 (e.g., invalid) and the control signal 10 is a read signal. When the data is restored to the volatile memory 220, the restored data may be transmitted to the CPU 100. The mode generating unit 310 may update the "I" bit of an address corresponding to the data restored to the volatile memory 220 from 1 to 0.

[0082] Moreover, if the "I" bit of the look-up table is 0 (e.g., valid) and the control signal 10 is a read signal, the mode generating unit 310 may transmit the input control signal 10 to the volatile memory 220 such that the data corresponding to an address can be output from the volatile memory 220 to the CPU 100 outside of the memory apparatus 200.

[0083] The first arbiter 320 and the second arbiter 330 may control a signal related to a data bus to prevent various signals from being mixed together in the data bus.

[0084] FIG. 4 is a flowchart depicting an example of a method for a memory apparatus. Data restoration from a non-volatile memory to a volatile memory in an on-demand fashion at ZPSTR releasing booting operation will be described below with reference to FIGS. 1 and 4.

[0085] The memory control unit 210, at operation 410, may receive a command for reading data corresponding to an input address, and then it may be determined, at operation 420, whether data corresponding to the input address is present in the volatile memory 220.

[0086] If it is determined, at operation 420, that the data corresponding to the input address is not present in the volatile memory 220, the memory control unit 210, at operation 430, may restore data stored in the non-volatile memory 230 to the volatile memory 220, and, at operation 440, may read data from the volatile memory 220 and transmit the read data to the CPU 100.

[0087] If it is determined, at operation 420, that the data corresponding to the input address is present in the volatile memory 220, the memory control unit 210 may read data from the volatile memory 220 and transmit the read data to the CPU 100.

[0088] FIG. 5 is a flowchart illustrating another example of a method for a memory apparatus.

[0089] Referring to FIGS. 3 and 5, when the memory apparatus 200 stays in suspend-to-RAM mode after ZPSTR releasing booting operation at operation 510, the decoder 212, at operation 520, may decode an address and a control signal which are input from the CPU 100. The restoration determining unit 214, at operation 530,

may read an "I" bit from the look-up table 215, which may correspond to the address, and, at operation 540, may determine whether the "I" bit indicates validity which represents that data corresponding to the address is present in the volatile memory 220.

[0090] If it is determined, at operation 540, that the data corresponding to the address is not present in the volatile memory 220, the mode generating unit 310, at operation 560, may generate control signals for the volatile memory 220, the non-volatile memory 230, the first arbiter 320, and the second arbiter 330.

[0091] Data may be transmitted from the non-volatile memory 230 to the volatile memory 220 under the control of the mode generating unit 310 at operation 570. Once the data is transmitted to the volatile memory 220, the volatile memory 220 may have the data corresponding to the input address, and thus the mode generating unit 310 may update the "I" bit corresponding to the input address in the look-up table 215 from 1 (e.g., invalid) to 0 (e.g., valid) at operation 580. The data restored to the volatile memory 220 may be loaded onto the data bus and transmitted to the CPU 100.

[0092] If the "I" bit indicates that the data corresponding to the input address is present in the volatile memory 220, the mode generating unit 310 may transfer the control signal to the volatile memory 220 at operation 550, and data according to the control signal may be loaded from the volatile memory 220 to the data bus and transmitted to the CPU 100 at operation 590.

[0093] Embodiments also may provide improved switching between a virtual machine, e.g., Xen, and a guest operating system (OS), e.g., Linux, Windows, MacOS, etc. A native OS, (e.g., Linux, Windows, MacOS, etc.) may be used in place of the virtual machine. Also, software simulation may be available. FIG. 6 is a flow-chart illustrating another example of a method 600 according to an embodiment. When setting up the guest OS, it may be done such that the virtual machine is informed of the area of a general storage or the volatile memory 220 in which the contents of the register 105 in the CPU 100 is stored (operation 610). The guest OS may be set to be terminated using suspend-to-RAM (operation 620).

[0094] The virtual machine may confirm that suspend-to-RAM is completed in the guest OS (operation 630), and may store all contents of a general volatile memory module which is currently used or the volatile memory 220 area in the memory apparatus 200 (operation 640). For reference, volatile memory 220 areas of the guest OS and the virtual machine may be partitioned. Data may be stored in a general storage unit using suspend-to-disk function of the virtual machine. Power-off operation which completely turns off power may be performed (operation 650). In response to power-on (operation 660), hibernation of the virtual machine may be restored (operation 670).

[0095] When restoration of the virtual machine is completed, the contents of the register 105 is restored from

general volatile memory module or the memory apparatus 200 corresponding to the guest OS volatile memory 220 area (operation 680) or general storage area. When an instruction is executed in the CPU 100, memory address access may be requested. At this time, the access may pass through a pseudo physical memory layer of the virtual machine using "page fault," and it can be known which memory area is requested. Restored and non-restored areas may be managed by the virtual machine with a table or a bitmap. When memory access to the non-restored area occurs, the memory apparatus 200 or virtual machine may read the area on a block-by-block basis, and can restore the area immediately.

[0096] A memory apparatus includes a volatile memory, a non-volatile memory, and a memory controller configured to restore data, according to an externally input signal, from the non-volatile memory to the volatile memory in an on-demand fashion during booting. The memory control unit may store data from the volatile memory to the non-volatile memory before power is disconnected from the volatile memory. In addition, the memory control unit may transfer data back into the volatile memory from the non-volatile memory. A lookup table may be used to correlate the addresses in the volatile memory to the data stored in the non-volatile memory.

[0097] The processes, functions, methods, operations, and/or software described above may be recorded, stored, or fixed in one or more computer-readable storage media that includes program instructions to be implemented by a computer to cause a processor to execute or perform the program instructions. The media may also include, alone or in combination with the program instructions, data files, data structures, and the like. The media and program instructions may be those specially designed and constructed, or they may be of the kind well-known and available to those having skill in the computer software arts.

[0098] Examples of computer-readable media include magnetic media, such as hard disks, floppy disks, and magnetic tape; optical media such as CD-ROM disks and DVDs; magneto-optical media, such as optical disks; and hardware devices that are specially configured to store and perform program instructions, such as read-only memory (ROM), random access memory (RAM), flash memory, and the like. Examples of program instructions include machine code, such as produced by a compiler, and files containing higher level code that may be executed by the computer using an interpreter. The described hardware devices is configured to act as one or more software modules in order to perform the operations and methods described above, or vice versa. In addition, a computer-readable storage medium may be distributed among computer systems connected through a network and computer-readable codes or program instructions may be stored and executed in a decentralized manner.

[0099] The memory module, which may include Flash/DRAM memory devices and/or memory controllers, may be included in various types of packages. For example,

the flash/DRAM memory devices and/or memory controllers may be embodied using packages such as Package on Packages (PoPs), Ball Grid Arrays (BGAs), Chip Scale Packages (CSPs), Plastic Leaded Chip Carrier (PLCC), Plastic Dual In-Line Package (PDIP), Die in Wafer Pack, Die in Wafer Form, Chip On Board (COB), Ceramic Dual In-Line Package (CERDIP), Plastic Metric Quad Flat Pack (MQFP), Quad Flatpack (QFP), Small Outline Integrated Circuit (SOIC), Shrink Small Outline Package (SSOP), Thin Small Outline (TSOP), Thin Quad Flatpack (TQFP), System In Package (SIP), Multi Chip Package (MCP), Wafer-level Fabricated Package (WFP), Wafer-Level Processed Stack Package (WSP), and the like.

[0100] The memory module, which may include the flash/DRAM memory devices and/or the memory controllers, may include memory cards. In this case, the memory controllers may be constructed to communicate with an external device for example, a host using any one of various types of interface protocols such as a Universal Serial Bus (USB), a Multi Media Card (MMC), a Peripheral Component Interconnect-Express (PCI-E), Serial Advanced Technology Attachment (SATA), Parallel ATA (PATA), Small Computer System Interface (SCSI), Enhanced Small Device Interface (ESDI), and Integrated Drive Electronics (IDE).

[0101] The flash memory devices may be non-volatile memory devices that can maintain stored data even when power is cut off. According to an increase in the use of mobile devices such as a cellular phone, a personal digital assistant (PDA), a digital camera, a portable game console, and an MP3 player, the flash memory devices may be more widely used as data storage and code storage. The flash memory devices may be used in home applications such as a high definition television (HDTV), a DVD, a router, and a Global Positioning System (GPS).

[0102] A number of example embodiments have been described above. Nevertheless, it will be understood that various modifications may be made. For example, suitable results may be achieved if the described techniques are performed in a different order and/or if components in a described system, architecture, device, or circuit are combined in a different manner and/or replaced or supplemented by other components or their equivalents. Accordingly, other implementations are within the scope of the following claims.

Claims

1. A method for switching between a virtual machine and a guest operating system (OS) in a computer, the computer configured to support a suspend-to-RAM function, the method comprising:

informing the virtual machine of an area in a volatile memory in which contents of a register, cor-

responding to the to the guest OS, is stored from the register (610), the volatile memory comprising at least one area;
 setting the guest OS to be terminated using suspend-to-RAM (620);
 confirming, by the virtual machine, that suspend-to-RAM is completed in the guest OS (630);
 storing, by the virtual machine, all contents of the area in the volatile memory (640);
 performing a power-off operation (650);
 performing a power-on function (660);
 restoring the virtual machine (670); and
 restoring the register contents to the register from the area in the volatile memory (680).

2. The method of claim 1, further comprising any one or more of the following steps:

- partitioning the area in volatile memory into a plurality of areas respectively corresponding to the guest OS and the virtual machine; and/or
- storing data in a general storage unit using a suspend-to-disk function of the virtual machine.

3. The method of claim 1, further comprising, in response to an instruction being executed in the computer, memory address access is requested.

4. The method of claim 3, further comprising passing the access through a pseudo physical memory layer of the virtual machine using a page fault.

5. The method of claim 1, further comprising any one or more of the following steps:

- managing, by the virtual machine, restored and non-restored areas with a table;
- and/or
- managing, by the virtual machine, restored and non-restored areas of the volatile memory with a bitmap.

6. The method of claim 1, further comprising, in response to memory access to a non-restored area occurring, reading the area on a block-by-block basis.

7. The method of claim 6, further comprising restoring the non-restored area.

8. A computer supporting a suspend-to-RAM function, the computer comprising:

- a central processing unit (CPU) comprising a register, the register comprising contents;
- a volatile memory comprising at least one area;
- a non-volatile memory;

a virtual machine;
 a guest operating system (OS) being configured
 to be terminated by suspend-to-RAM; and
 a memory controller configured to:

5

switch between the virtual machine and the
 guest (OS); and
 inform the virtual machine of an area in a
 volatile memory in which contents of a reg-
 ister, corresponding to the to the guest OS,
 are stored from the register, wherein the vir-
 tual machine is configured to:

10

confirm that suspend-to-RAM is com-
 pleted in the guest OS,
 store all contents of the area in the vol-
 atile memory,

15

wherein the computer is configured to:

20

perform a power-off operation, and
 performing a power-on function,

wherein the memory controller is further config-
 ured to:

25

restore the virtual machine, and
 restore the register contents to the register
 from the area in the volatile memory.

30

9. The computer of claim 8, wherein the memory con-
 troller is further configured to partition the area in
 volatile memory into a plurality of areas respectively
 corresponding to the guest OS and the virtual ma-
 chine.

35

10. The computer of claim 8, further comprising a gen-
 eral storage unit configured to store data using a
 suspend-to-disk function of the virtual machine.

40

11. The computer of claim 8, wherein the computer is
 further configured to execute and instruction to re-
 quest memory address access.

12. The computer of claim 11, wherein the virtual ma-
 chine is further configured to pass the access
 through a pseudo physical memory layer of the vir-
 tual machine using a page fault.

45

13. The computer of claim 8, wherein the virtual machine
 is further configured to manage restored and non-
 restored areas of the volatile memory with a table.

50

14. The computer of claim 8, wherein the virtual machine
 is further configured to manage restored and non-
 restored areas of the volatile memory with a bitmap.

55

15. The computer of claim 8, wherein the memory con-

troller is further configured to, in response to memory
 access to a non-restored area occurring, read the
 area on a block-by-block basis; wherein the memory
 controller is preferably further configured to restore
 the non-restored area.

FIG. 1

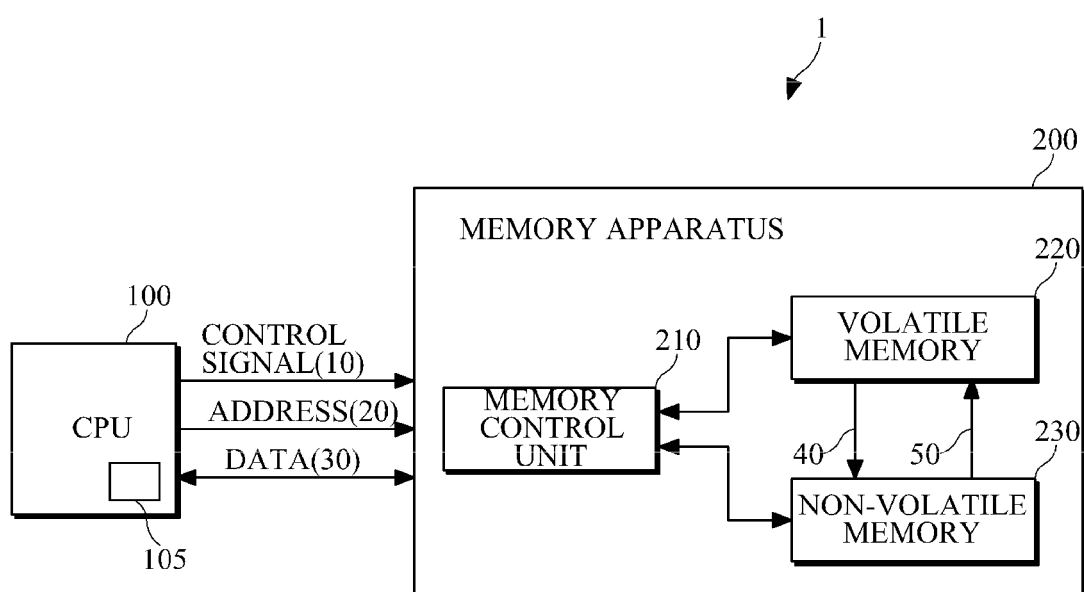


FIG. 2

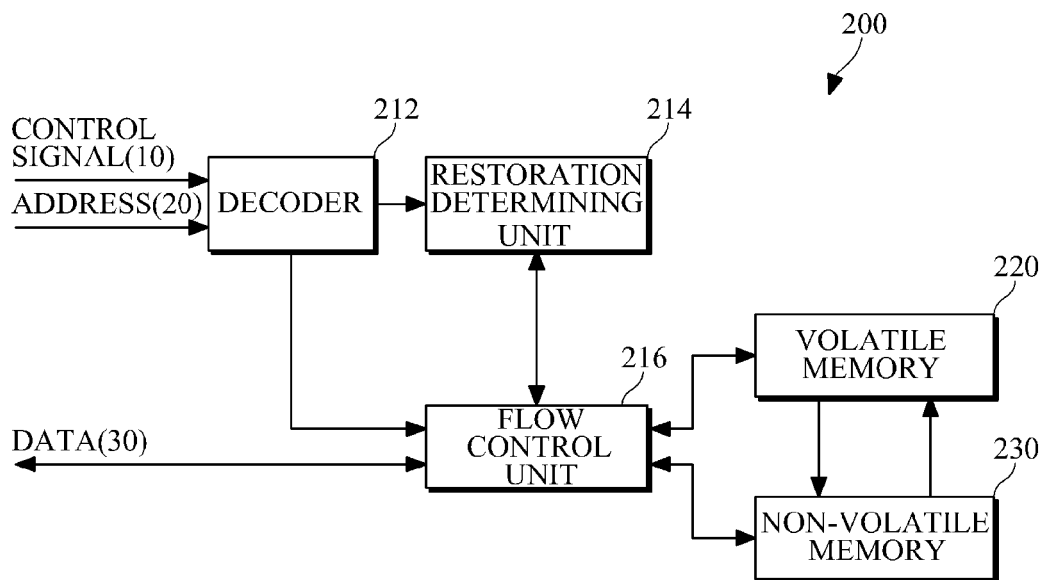


FIG. 3

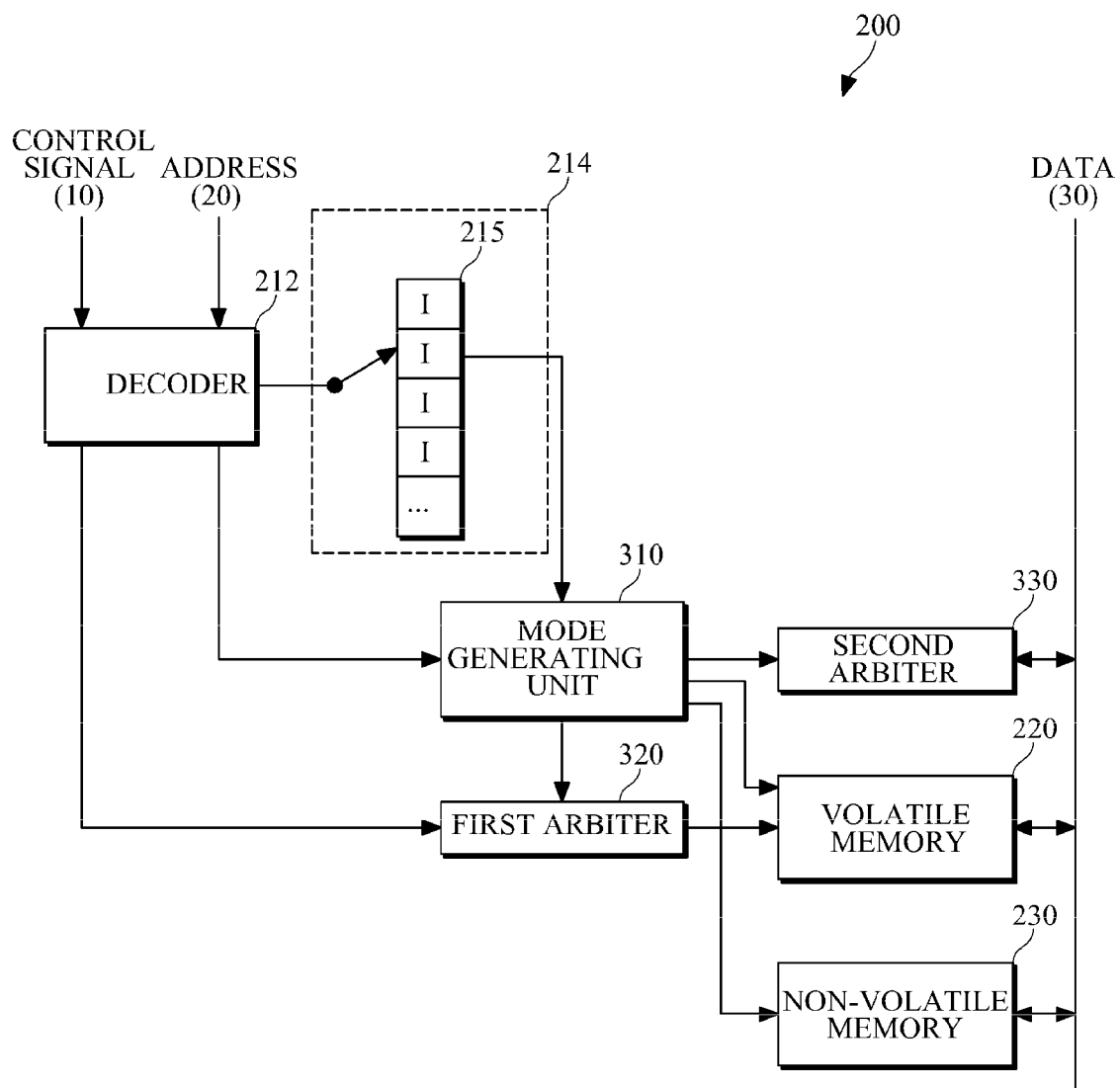


FIG. 4

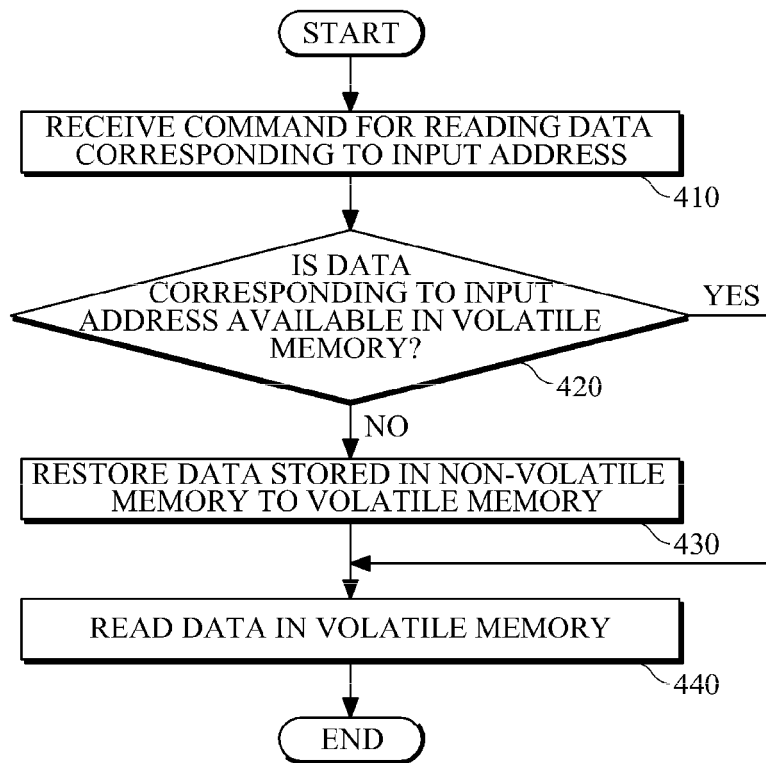


FIG. 5

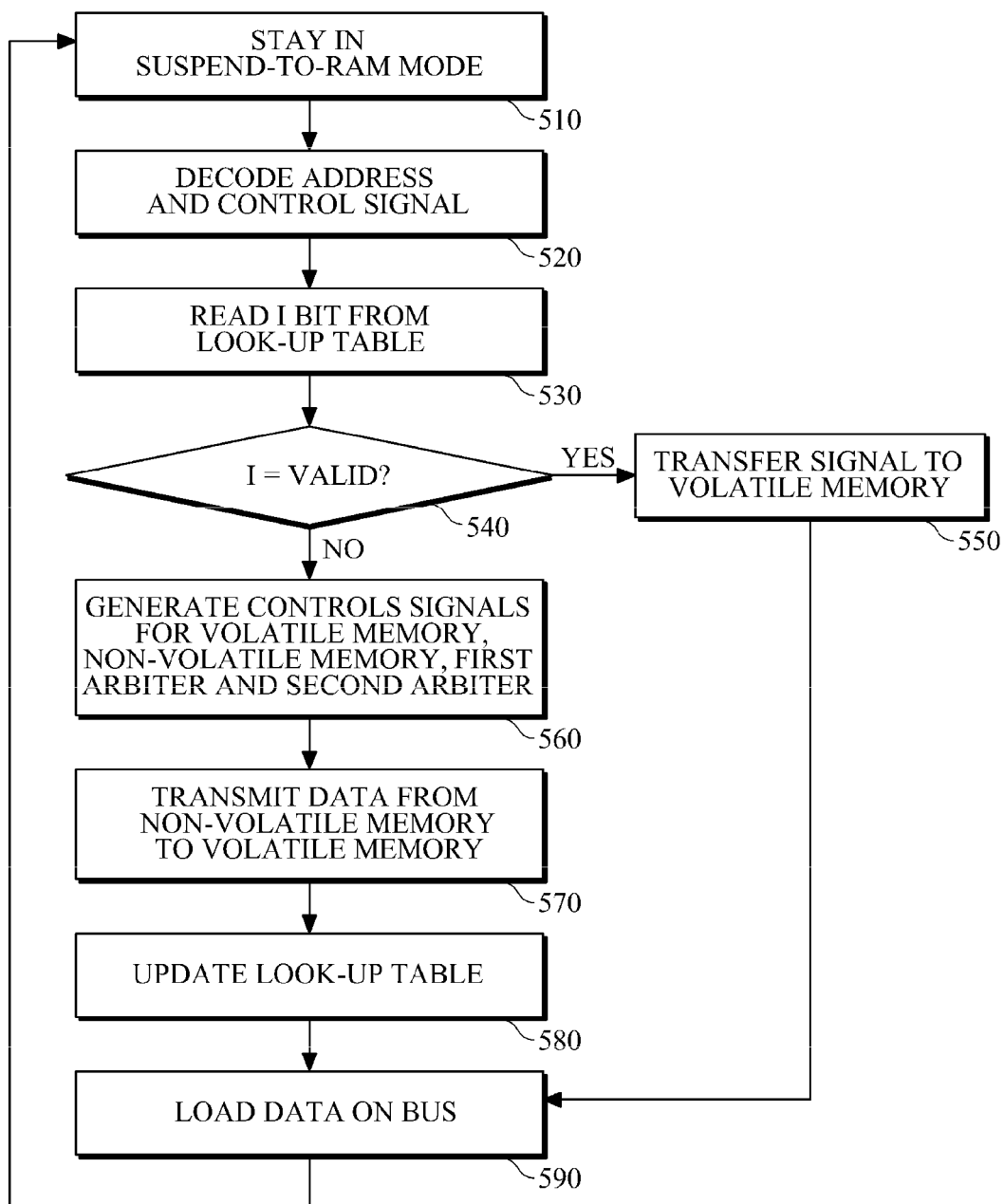
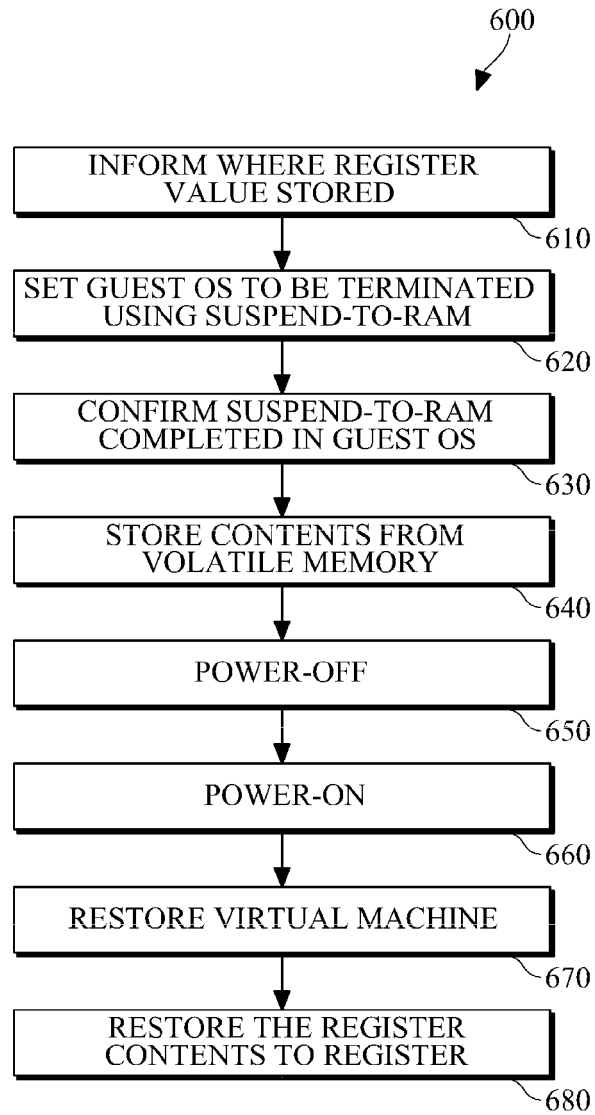


FIG. 6





EUROPEAN SEARCH REPORT

 Application Number
 EP 10 16 3389

DOCUMENTS CONSIDERED TO BE RELEVANT			
Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (IPC)
X	US 7 356 677 B1 (RAFIZADEH SCHUMANN [US]) 8 April 2008 (2008-04-08) * column 4, line 33 - column 6, line 5 * * column 13, line 24 - column 15, line 48 *	1-15	INV. G06F13/16
A	US 2008/189538 A1 (KING BRIAN DOUGLAS [US] ET AL) 7 August 2008 (2008-08-07) * the whole document *	1-15	
A	US 2002/087816 A1 (ATKINSON LEE W [US] ET AL) 4 July 2002 (2002-07-04) * figure 1 * * paragraph [0020] - paragraph [0022] * * paragraph [0057] * * paragraph [0073] - paragraph [0080] *	1-15	
A	US 2008/005541 A1 (HASE TATSUO [JP] ET AL) 3 January 2008 (2008-01-03) * paragraph [0007] - paragraph [0015] *	1-15	
A	Intel: "Intel NAND Flash Memory Intel Turbo Memory" 31 May 2007 (2007-05-31), XP002591336 Retrieved from the Internet: URL: http://download.intel.com/design/flash/nand/turbomemory/whitepaper.pdf [retrieved on 2010-07-09] * figure 7 * * page 6, column 1, last paragraph *	1-15	TECHNICAL FIELDS SEARCHED (IPC) G06F
The present search report has been drawn up for all claims			
Place of search Munich		Date of completion of the search 20 July 2010	Examiner Ghidini, Mario
CATEGORY OF CITED DOCUMENTS X : particularly relevant if taken alone Y : particularly relevant if combined with another document of the same category A : technological background O : non-written disclosure P : intermediate document T : theory or principle underlying the invention E : earlier patent document, but published on, or after the filing date D : document cited in the application L : document cited for other reasons & : member of the same patent family, corresponding document			

 1
 EPO FORM 1503 03.82 (P04C01)

**ANNEX TO THE EUROPEAN SEARCH REPORT
ON EUROPEAN PATENT APPLICATION NO.**

EP 10 16 3389

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report.
The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

20-07-2010

Patent document cited in search report		Publication date	Patent family member(s)	Publication date
US 7356677	B1	08-04-2008	NONE	

US 2008189538	A1	07-08-2008	AU 2008214236 A1	14-08-2008
			CA 2675523 A1	14-08-2008
			CN 101606128 A	16-12-2009
			EP 2109818 A1	21-10-2009
			JP 2010518512 T	27-05-2010
			WO 2008097695 A1	14-08-2008

US 2002087816	A1	04-07-2002	US 2003131206 A1	10-07-2003

US 2008005541	A1	03-01-2008	JP 2007334383 A	27-12-2007
